Assignment 1 Clarifications

- Q1b: State whether having the code snippet in your code will cause compilation errors. Explain why or why not.

- Q3: For the code snippets, how many times is the `printf` statement executed? Briefly explain (up to 3 sentences).

- Q4: Write a function to draw an isosceles triangle of a given height N using for/while loop.

- Hand in hard copy before Wednesday class in CSIL (across from main CS office)

Exam Hardship

- Three or more finals scheduled within 24 hours
  - Let me know via email ASAP if you haven't already
  - mochen@cs.sfu.ca

- Exam at one location (e.g. Burnaby) followed immediately by exam at another location (e.g. Surrey).
  - Math 152 Final will be in Burnaby
  - Please check with and thank Prof. Jungic

  - Please keep up with both courses
  - Last two lectures will be review sessions

Array Comparison, Strings and Loops

CMPT 125
Jan. 14

Lecture 5

Today

- Array Comparison
- Strings
- Nested Loops

Array Comparison

Puzzle: What’s wrong with this code?

```
int main ( ) {
    int password[3] = {1,2,3};
    int answer[3];

    for (int i = 0; i < 3; i++) {
        printf("Enter digit %d: ", i+1);
        scanf("%d", answer+i);
    }
    if (password != answer) {
        printf("Incorrect password!\n");
    }
}
```

probably a bug

compares the values of the pointers, not the array elements

Array Comparison

- Write a function to compare two arrays
- Array parameters passed by base address
    - Style points: use `int arr[]` instead of `int *arr`

```
int arrCompare(int A[], int B[], int length) {
    for (int i = 0; i < length; i++) {
        if (A[i] < B[i]) {
            return -1;
        } else if (A[i] > B[i]) {
            return 1;
        }
    }
    return 0;
}
```

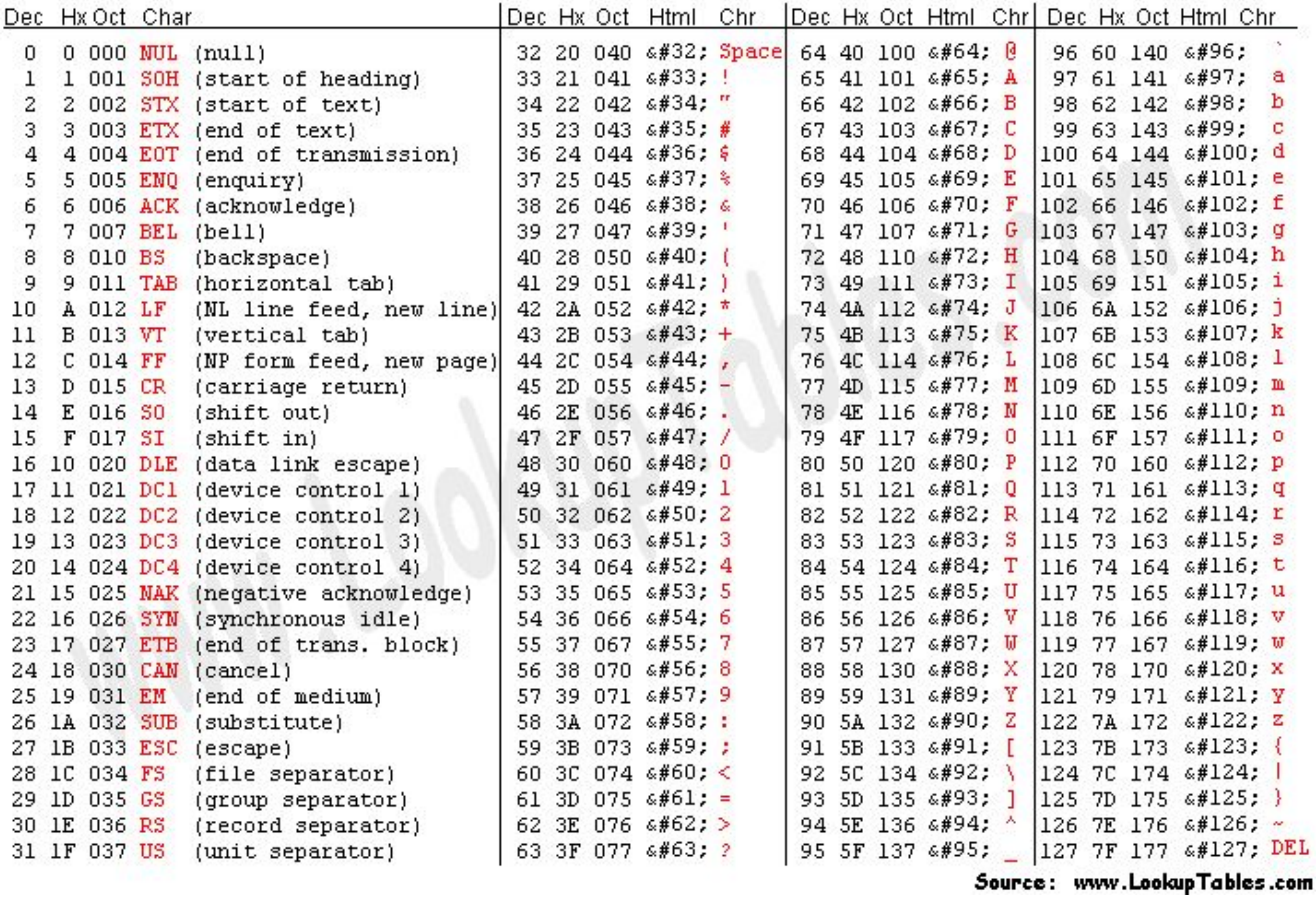
array bounds passed separately

Arrays of `char`

- type `char` is 1 byte per element
    - traditionally to hold one ASCII character
    - an array of `char` is a string!
- end of string terminated by *null char*: `'\0'`

```
int main ( ) {
    char msg[10] = "ur n00b!";
    printf("%s\n", msg);
}
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
msg[10]:	'u'	'r'	' '	'n'	'0'	'0'	'b'	'!'	'\0'	
	117	114	32	110	48	48	98	33	0	



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

String Comparison

```
#include <stdio.h>
#include <string.h>

int main ( ) {
    char password[4] = "abc";
    char answer[4];

    printf("Enter 3-character code: ");
    scanf("%s", answer);
    if (strcmp(password, answer) != 0) {
        printf("Incorrect password!\n");
    }
}
```

not `&answer` because `answer` is a pointer!

C library function to do string comparisons:

- 0 means equal
- < 0 means first < last
- > 0 means first > last

Common String Functions

`int strlen(char s[])`

- returns the length of the string
- counts characters until null terminator
- Q: What happens if there is no null terminator?

`void strcpy(char dest[], char src[])`

- copies the string `dest[]` ← `src[]`
- Q: What *must* be true about `dest[]`?

String I/O

Input

- `scanf("%s", str);`
- `scanf("%[^\n]s", str);`
- `gets(str);`

Also dangerous

Dangerous

Output

- `printf("%s", str);`
- `puts(str);`

Nested Loops

- It is possible to include any sequence of statements within a loop body including:
  - calculations
  - function calls
  - if statements
  - other loops
- Just like you did in Python!

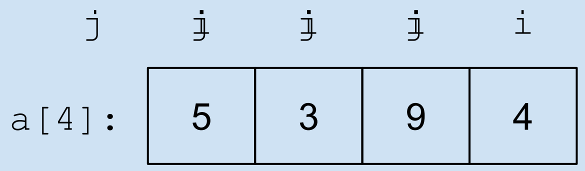
Classic Problem: Write a function that scans an array of `int`. It returns 1 if and only if two of the elements are the same, 0 otherwise.

Classic Solution

```
int dup_chk(int a[], int length) {
    int i = length;
    while (i > 0) {
        i--;
        int j = i - 1;
        while (j >= 0) {
            if (a[i] == a[j]) {
                return 1;
            }
            j--;
        }
    }
    return 0;
}
```

Simulation:

`dup_chk(a[4], 4):`



These statements run the most frequently in the worst case

- What is the worst case?
- How many times when length = 4?

Another Performance Measure

- Often consider the *worst-case* behaviour as a benchmark
    - make guarantees about code performance under all circumstances
- Can predict performance by counting the number of steps required by algorithm in the worst case
    - Derive total steps (T) as a function of input size (N)

Analysis

```
      int dup_chk(int a[], int length) {
1         int i = length;
N + 1     while (i > 0) {
N             i--;
N             int j = i - 1;
i + 1         while (j >= 0) {
i                 if (a[i] == a[j]) {
                      return 1;
                  }
i                 j--;
              }
          }
1         return 0;
      }
```

Q. What is N?

- The number of elements in the array

Outside of loop: 2 (steps)

Outer loop: $3N + 1$

Inner loop: 3`i`$+ 1$ for all possible `i` from 0 to $N - 1$.

$= 3/2\ N^2 - 1/2\ N$

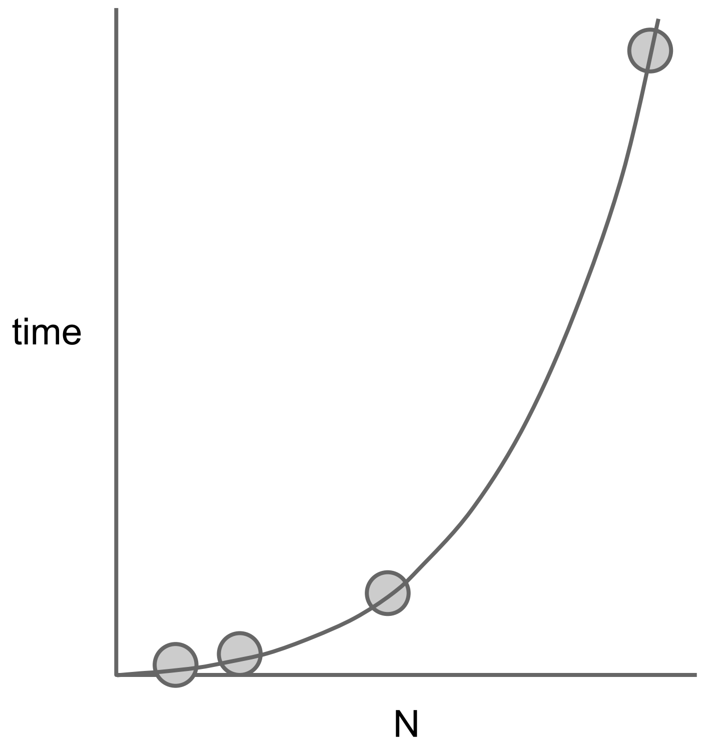
Grand total $= 3/2\ N^2 + 5/2\ N + 3$

A *quadratic* function!

Empirical Measurement

- Another graph - a quadratic this time!
- Confirms predictions: *doubling* (x2) the input size leads to **quadrupling** (x4) the running time

N	time (in ms)
10,000	89
20,000	365
40,000	1,424
100,000	9,011



2D Maximum Density Problem

Problem: Given a 2-dimensional array (NxN) of integers, find the 10x10 patch that yields the largest sum

Applications:

- Resource management and optimization
- Finding brightest areas of photos

Algorithm / Code?

- Simple approach: Try all possible positions for the upper left corner
  - (N-9)x(N-9) of them
  - use a nested loop
- add each patch using a 10x10 nested loop
- A *brute-force* approach!
  - Generate a possible solution [naively]
  - Test it [naively]

In C

```
int max10by10(int a[N][N]) {
    int best = 0;
    [redacted]
            int total = 0;
            for (int row = u_row; row < u_row+10; row++) {
                for (int col = u_col; col < u_col+10; col++) {
                    total += a[row][col];
                }
            }
            best = max(best, total);
        }
    }
    return best;
}
```

In C

```
int max10by10(int a[N][N]) {
    int best = 0;
    for (int u_row = 0; u_row < N-9; u_row++) {
        for (int u_col = 0; u_col < N-9; u_col++) {
            int total = 0;
            for (int row = u_row; row < u_row+10; row++) {
                for (int col = u_col; col < u_col+10; col++) {
                    total += a[row][col];
                }
            }
            best = max(best, total);
        }
    }
    return best;
}
```

x(N-9)

x(N-9)

x10

11

10

10

Approximate Method:

Count the *barometer instructions*, the instructions executed most frequently. Usually, in the innermost loop.

Innermost loop: 11 + 10 + 10 = 31 ops

Total = 31 **x10 x(N-9)** **x(N-9)** **=$310N^2$**